ECE/CSE576 Final Report - Optimizing "A Keypoint-based Global Association Network for Lane Detection"

Harsha Vardhan, Moses Prasad Varghese*, Ryan Ching
Department of Electrical and Computer Engineering
Department of Mechanical Engineering*
University of Washington
Seattle, WA, USA
{harshav,mosespv,chingry}@uw.edu

Abstract

Our motivation for choosing this paper was because of its potential real world use in self-driving cars. The original paper proposes a Global Association network using keypoint estimation. Key points refers to lane points with features that denote the edge of a lane. The machine learning aspect is in using convolutional neural networks (CNN) to extract the features of keypoints in a lane. As such, we decided to focus on reproducing this paper's results using the backbone, as well as implementing a new backbone to compare results. Our goal of this project was to improve the existing model of keypoint detection, by improving the backbone architecture of this paper from ResNet to ResNext. In this paper, we will discuss the optimization we implemented as well as how we were able to reproduce the original results of the paper. We compare the results of ResNext to ResNet to see how the F1 score changes.

1 Related Works

Our new implemented backbone is the ResNeXt architecture as highlighted in [4].Our prediction is that this implementation would improve the runtime while allowing for a more simplified implementation, as it relies on less hyperparameters. The paper also highlights increased performance when compared to ResNeXt, although the weights for first 10 layers were trained on ImageNet and then with our TuSimple dataset. We used the ImageNet weights to incorporate transfer learning to increase accuracy.

2 Model Description and Methodology

Our implementation and planned optimization relies entirely on the original paper, [3]. In summary, this paper aims to use the input image passed through the backbone (ResNet[1]) and feature pyramid network (FPN [2]) to extract features and determine keypoints. An example of gathering keypoints from [3] is shown in the figure below.

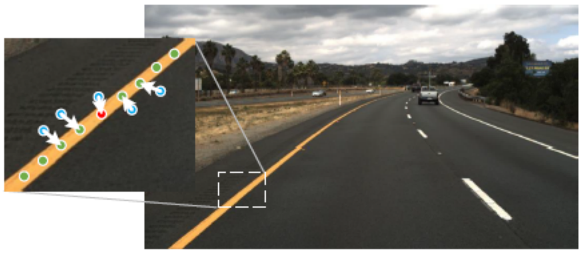


Figure 1. Highlighted Keypoints from original image. Taken directly from "A Keypoint-based Global Association Network for Lane Detection" Paper

These features are passed through two different heads, both consisting of different sequential CNNs to create maps.
To create the confidence map, the keypoints as output of the FPN is passed through a Lane Feature Aggregator, which takes all key points and predicts the estimated offset between a chosen start point and all other keypoints. If the position of these offset values falls within some distance of the keypoint, the keypoints are considered to be part of the same line. This is then passed through a keypoint head to produce a confidence map. This confidence map is used to evaluate performance of predictions.
On the other hand, the output of the FPN is also passed through an offset head, a series of Convolutional neural network layers to calculate the estimated offset from a chosen starting point for each lane line and all other keypoints that may be a part of the same line. In defining the lane lines when inference is run, all keypoints are offset by their estimated values in the offset map. If these offset value is in the neighborhood (for our case, the offset value falling within 25 pixel of the starting point), we consider this part of the same line.

3 Implementation

We implemented the backbone network, self-attention layer and the FPN modules of the paper [3], while retaining the other modules such as trainer, data readers and loaders, evaluation scripts from the implementation of the original authors of [3]. This allowed us to reproduce the results of the paper [3], and gave us the potential to implement and experiment with better backbone networks. In the section below, we will be implementing and attempting to improve is replacing the backbone from ResNet to ResNext. The main objective was to compare performance of the two different architectures in terms of accuracy. The main advantages of ResNeXt is the grouping of convolutional layers, meaning the inference should be computed faster. The accuracy was defined based on the correctly predicted key points. We give a 20 pixel radius to determine the correctly predicted keypoints with the ground truth. Then we find the true positive of the lane by giving the accuracy as 0.85. Based on this analysis we calculate the F1 score of our ResNeXt model and the original ResNet model.

3.1 Technical Description of the new backbone

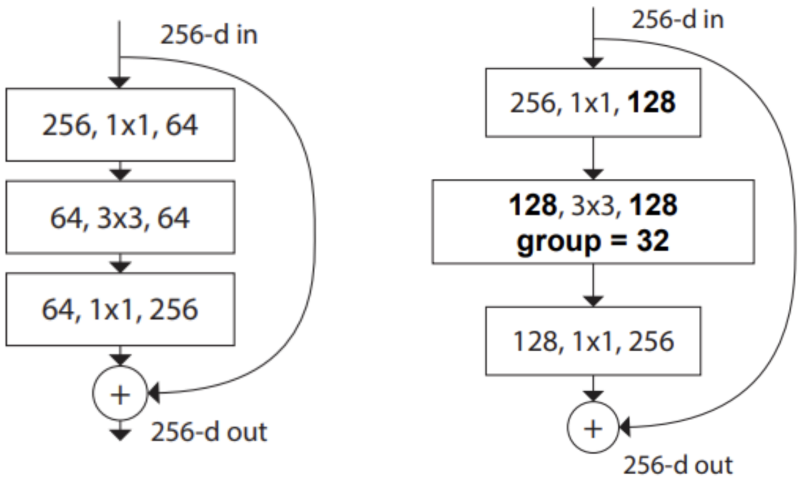


Figure 2. Bottleneck block of ResNet (Left) vs ResNeXt (Right) Architecture

In case of the ResNet bottleneck on the left, we can see that a 256 channel input is downsampled to 64 channels along with keeping the feature map size the same, with the help of 1x1 convolutions. Then 3x3 convolutions are performed on the downsampled dimensions before upsampling the channels to 256 again using 1x1 convolution kernels. The resulting feature map is added with the input forming the skip/shortcut connections of the bottleneck block added to carryover gradients preventing gradient exploding/vanishing issues. The same outcome (in-terms of tensor manipulations and receptive field) is achieved using the bottleneck block of ResNeXt but with fewer learnable parameters and greater parallelization from the grouped convolutions involved. This allows us to train and deploy larger backbone models like ResNeXt-50 and ResNeXt-101.

4 Discussion

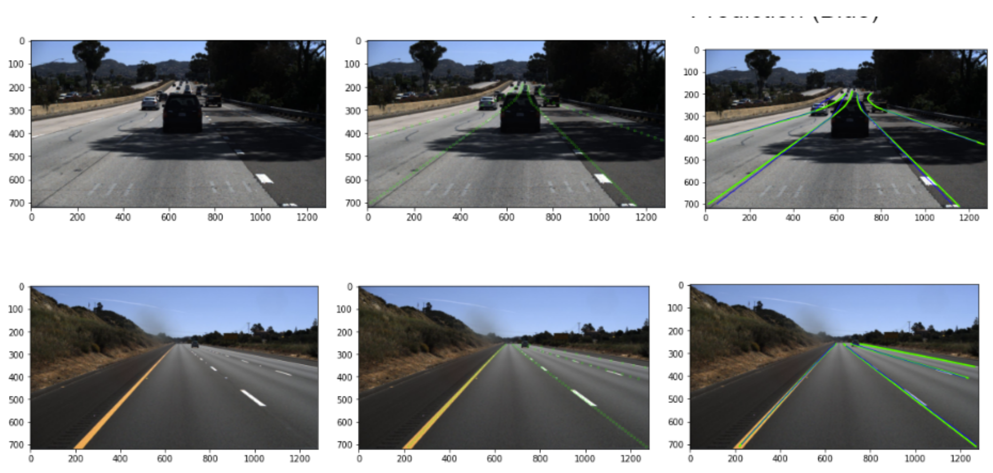
4.1 Experimental results

Table 1: Experimental results

Backbone Architecture	F1 Score on TuSimple Dataset
ResNet-34 - Pretrained Weights of GANet-M	97.59
ResNet-101 : pretrained weights of GANet-L	97.40
ResNeXt-50 : finetuned on ImageNet weights	95.33
ResNeXt-101 : finetuned on ImageNet weights	91.71

The table above summarizes the experiments we were able to conduct after implementing the backbone, self-attention and Feature Pyramid Network(FPN [2]) modules of the [3]'s full architecture, along with implementing the new ResNeXt backbones. The first two ResNet F1 scores were taken by reproducing the papers results by running inference on the weights provided in the repository. The ResNext F1 scores was our modified implementation.

We calculated the accuracy exactly how it was implemented in the paper, as our evaluation metric must be the same to determine which model has better performance. The model is provided in the TuSimple code base. However, our accuracy ended up being lower than the paper's original results. This could be caused by fusing the convolutional layers lowering the accuracy as a tradeoff for decreasing runtime. The below figures show the results from the analysis from the ResNet and ResNeXt models. The first row of images shows for ResNet34, second row for ResNet101, third row for ResNeXt50 and fourth row for ResNeXt101. The first column represents the original image, second column represents the expected lane with original image and the third column represents the comparison between the ground truth in green and prediction in blue



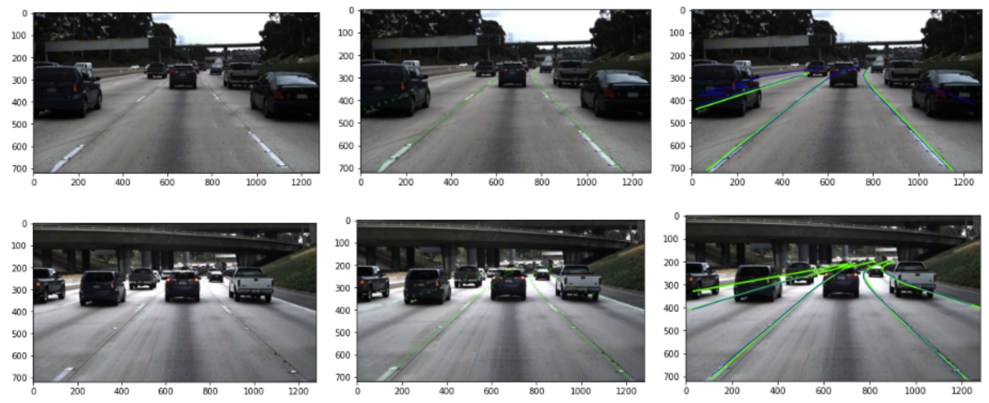


Figure 3. Results for ResNet and ResNeXt models

5 Future Work

As this paper relied had many components, there are many options for future optimization as well as methods we can improve in our experiment. However, due to time constraints we were unable to test out all of the planned implementations. We can further test the performance of different architectures such as EfficientNet and vision transformers. The process of implementation would be similar to the steps taken to implement ResNeXt. There are more evaluation metrics we can implement to test our prediction that ResNeXt would have faster inference speeds, as simply tracking runtime would not be sufficient as there are other variables affecting runtime.
Additionally, the paper mentions how the output stride set to 1 can make regression calculations difficult due to the large absolute value of the offsets. Our original project proposal was to fix this by allowing for multiple regression, however we determined that this would be hard to measure improvement, as this regression is used as an evaluation metric rather than for keypoint estimation. Increasing the output stride woluld lead to downsampling of the image more, which could decrease performance. It would be interesting to observe hwo the performance vs run time changes.
Looking through the repository for this paper, there are also unimplemented modules such as velocity evaluation which seemed like interesting topics. However, this section was not mentioned in their research paper and was unrelated to the rest of their paper. A final option for future improvement would be to implement and evaluate other lane estimation methods, such as anchor based methods which regress keypoint by keypoint to estimate the lane. The inference time would be longer but performance could also improve.

6 Conclusion

In our project, we first used the original backbone architecture ResNet-34 and ResNet-101 each with the Feature Pyramid Network (FPN) module implemented in the paper to run the model. Then we developed our new model ResNeXt-50 and ResNeXt-101, an updated architecture of the ResNet model along with the FPN module and compared the F1 score using the predicted true positive lanes from the Tusimple test dataset. We observed that accuracy using ResNeXt decreased when compared to the papers original results, however the inference time difference was not calculated.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[2] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[3] Jinsheng Wang, Yinchao Ma, Shaofei Huang, Tianrui Hui, Fei Wang, Chen Qian, and Tianzhu Zhang. A keypoint-based global association network for lane detection. 2022.

[4] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016.